IMPLEMENTACIÓN DE GRAFCET SOBRE ARDUINO APLICANDO LÓGICA BINODAL

Jovanny Duque

Institución Universitaria ITSA
Soledad, Colombia

Jorge Duarte, Rafael Ramírez

Universidad del Atlántico
Barranquilla, Colombia

Resumen

El propósito de esta investigación es presentar una metodología que permita implementar sobre la plataforma de Arduino, una solución de automatización previamente simulada en GRAFCET, sobre la base de los teoremas de la lógica binodal. A través de este modelo se pueden superar la dificultad que representa llevar a la práctica un automatismo simulado, cuando no se cuenta con lineamientos lógicos que entrelacen estos dos contextos; y como segundo aspecto a resolver, está el hecho que los dispositivos electrónicos más comunes para implementar la solución encontrada, resultan de alto costo, como en el caso de los autómatas programables. Como herramientas de diseño, el proyecto integra el formalismo GRAFCET, la plataforma Arduino y la lógica binodal; el GRAFCET jugó el papel de lenguaje gráfico de programación para representar los sistemas secuenciales de eventos discretos, la lógica binodal aportó el modelo matemático que describe el estado de activación o desactivación de las etapas y Arduino, la económica plataforma electrónica, basada en software de código abierto, sobre la que se ejecuta la programación. Se ha probado que cualquier automatismo planteado bajo la descripción de uno o varios GRAFCET´s pueden ser traducido al código de Arduino sin pérdida de información, usando el modelo matemático generado desde la lógica binodal.

Palabras clave: GRAFCET; lógica binodal; Arduino

Abstract

The purpose of this investigation is to present a methodology that allows engineering students to implement on the Arduino platform any automation solution previously simulated in GRAFCET, based on the theorems of binodal logic. Through this model, the difficulties of carrying out a

simulated automatism practice can be overcome when there are no logical guidelines that intertwine these two contexts; and as a second aspect to be solved, there is the fact that the most common electronic devices to implement the solution found are high cost, as in the case of Programmable Logic Controller. As design tools, the project integrates the GRAFCET formalism, the Arduino platform, and binodal logic; GRAFCET played the role of graphical programming language to represent sequential systems of discrete events, binodal logic provided the mathematical model that describes the activation or deactivation status of the steps and Arduino, the economical electronic platform, based on code software open, on which the schedule is executed. It has been proven that any automation raised under the description of one or more GRAFCET's can be translated into the Arduino code without loss of information, using the mathematical model generated from the binodal logic.

Keywords: GRAFCET; binodal logic; Arduino

1. Introducción

Uno de los retos que debe enfrentar un desarrollador de equipos y maquinarias, ya sea que trabaje en el entorno industrial o un estudiante de pregrado, es hacer que su solución tecnológica sea posible implementarla al menor costo, sin prescindir de la confiabilidad. Cuando se trata de proyectos de automatización, ya existen equipos como los Autómatas Programable (PLC`s), con un hardware y software aceptado por la industria, cubriendo las necesidades de confiabilidad, seguridad y fácil programación, sin embargo, presentan el inconveniente de su alto costo. Actualmente existe la plataforma Arduino que ofrece microcontroladores, capaces de ejecutar secuencias, igual que lo hace un PLC, pero a una mucho mas economucos y fáciles de implementar. Por otra parte, estudiantes de áreas afines con la automatización, presentan dificultad para llevar a la realidad sistemas que han simulados previamente, por no contar con un modelo matemático o lineamientos metodológicos que les permitan relacionar estos dos ambientes.

En este trabajo se expone una metodología que permite implementar en el lenguaje “Programming” de Arduino, soluciones de automatización modeladas en GRAFCET, aplicando los teoremas de la lógica binodal.

Dentro de los enfoques que se han planteado para traducir GRAFCET o Redes de Petri a código de procesamiento, se destacan los trabajos de Schürenberg (2015), quien desarrolla la transformación de las especificaciones de control basadas en GRAFCET a un equivalente implementado con la norma IEC 61131-3 (2013). Este avance se apoyó los hallazgos realizados por Schumacher, (2013) (2014), que presentó un método para la generación automática del código a partir del GRAFCET mediante un exhaustivo modelado como una (CIPN) Red de Petri de control interpretada. Por su parte Giraldo, et al. (2019), concreta el desarrollo de “GRAFINO”, un software que permite obtener el código equivalente de Arduino a partir de un editor de GRAFCET.Probos (2011), propone un método para obtener una máquina de Mealy equivalente a partir de un GRAFCET, sin pérdida de semántica de información. En su propuesta Philippot (2010), presenta un enfoque para convertir un automatismo secuencial, a un modelo matemático equivalente, haciendo los enlaces semánticos entre una herramienta normalizada como GRAFCET y la estructura de un autómata

programable. En el enfoque propuesto por Borges P, et al. (2010), se tradujo una especificación SFC a código de lenguaje de programación C considerando el comportamiento del sistema en el dispositivo controlador. Aplicando un enfoque sistemático, Oriol (2008), propuso traducir las especificaciones de GRAFCET a un código equivalente para microcontroladores programados en C. Gi (2004), presentó un método para generar automáticamente código Ladder (LD) a partir de una (CPN) Control Petri Net, como resultado aparecen funciones de estado que pueden ser implementadas en un dispositivo programable. Villena, R. (2001). Realiza la ampliación y adaptación de la Teoría Binodal a automatismos industriales con PLCs.

2. Materiales y método

En esta sección se describirán un método práctico y sistemático para la traducción de un GRAFCET a código de programación "Processing" usando la teoría del binodo. A continuación, se amplían estas dos herramientas.

2.1 GRAFCET

Es una herramienta gráfica para la descripción del comportamiento de sistemas secuenciales de eventos discretos evolucionada a partir de las redes de Petri, contenida por la norma IEC 60848 (2013).

El GRAFCET de cualquier automatismo puede ser elaborado con los siguientes elementos: Etapas iniciales, etapas normales, transiciones, receptividades, saltos, reenvíos y acciones asociadas a las etapas (Figura 1).

Sin importar la complejidad del automatismo, es posible describirlo usando una combinación de las estructuras secuenciales básicas, como la estructura lineal, estructura alternativa OR (Divergencia / Convergencia) o estructura simultanea AND (Divergencia / Convergencia).

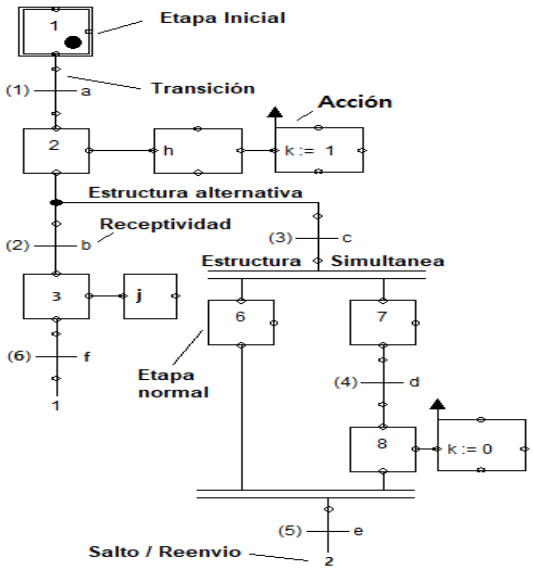

Figura 1. Estructuras y elementos del GRAFCET

2.2 Lógica binodal

La automatización de ciertos mecanismos puede volverse una tarea laboriosa y compleja. Ramos (1978), propone una estrategia rápida e intuitiva usando la lógica binodal. Allí, se genera un gráfico representativo de la dinámica del sistema, similar al usado en el lenguaje GRAFCET, y se obtienen directamente ecuaciones lógicas que describen el comportamiento del sistema aplicando teoremas binodales. Para entender el método es necesario conocer los siguientes conceptos:

Binodo: Estado o situación en el que puede presentarse un dispositivo, el binodo que tendrá dos posibles estados, S (activo) o $\bar{S}$ (inactivo) que serán disjuntos y complementarios.
v.d.a: Variables de activación del binodo.
v.d.d: Variables de desactivación del binodo.

La Figura 2 representa la activación o desactivación de un binodo, al lado izquierdo de S se están las (v.d.a) del binodo M_i y al lado derecho las P_j corresponde a las (v.d.d) que permiten conmutar al binodo a los dos estados mencionados:

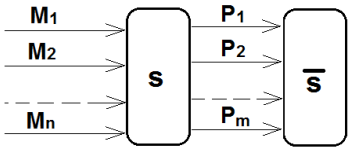

Figura 2. Activación y desactivación del binodo

Las expresiones matemáticas que representa la activación o desactivación de un binodo serán:
$\boldsymbol{S_{(t)} = \left[(S + \sum_{i=1}^{n} M_i) * \prod_{j=1}^{m} \overline{P_j}\right]}$ (1)

La ecuación (1) se puede extrapolar para definir el comportamiento de una etapa en GRAFCET, estas se comportan de forma igual que un binodo, teniendo solo dos posibles estados (activada o desactivada).

Como resultado de esta comparación, el estado de activación o desactivación de una etapa en GRAFCET se define de la siguiente manera:
$\boldsymbol{E_{(i)} = \left[\left(E_{(i)} + \sum_{i=1}^{n} E_{i-1} * \mathbf{t}_i\right) * \prod_{j=1}^{m} \overline{E}_{i+1}\right]}$ (2)
$\boldsymbol{E_{(i)}}$: Es el valor actual de la variable de la etapa i
$\boldsymbol{E_{(i+1)}}$:Es el valor actual de las variables de las etapas que suceden (aguas abajo) a $E_{(i)}$
$\boldsymbol{E_{(i-1)}}$:Es el valor actual de las variables de las etapas que preceden (aguas arriba) a $E_{(i)}$
$\mathbf{t_i}$: Es el valor de la transiciones i que preceden a $E_{(i)}$
n: número de transiciones que preceden a $E_{(i)}$
m: número de transiciones que suceden a $E_{(i)}$
$\sum_{i=1}^{n} E_{i-1} * \mathbf{t}_i$: Condición de activación es cualquier situación en la que una etapa precedente esté activa y se cumpla su receptividad, haciendo que se active la etapa $E_{(i)}$ y se desactiven la etapa o etapas E_{i-1} simultáneamente.

El algoritmo de la ecuación (2) es posible codificarlo al lenguaje "Processing" para implementarlo en tarjetas Arduino, usando los operadores booleanos de producto (AND) (**&**), suma (OR) (**|**) y negación (NOT) (**~**), como se aprecia en el caso estudiado en la sección 2,5.

2.3 modelo de conversión GRAFCET a un modelo matemático equivalente

La solución a un problema de automatización, inicia recopilando la mayor información y detalles posibles del funcionamiento deseado del proceso/sistema, a esta fase del proceso la llamaremos "Especificaciones del sistema" , una vez se tenga conocimiento general de la situación se procede a representar en forma gráfica las especificaciones en términos de etapas y transiciones, respetando la sintaxis de GRAFCET, a partir de allí, es posible aplicar al GRAFCET el modelo matemático a cada etapa y obtener el código equivalente para cargar al Arduino, como lo ilustra la Figura 3.

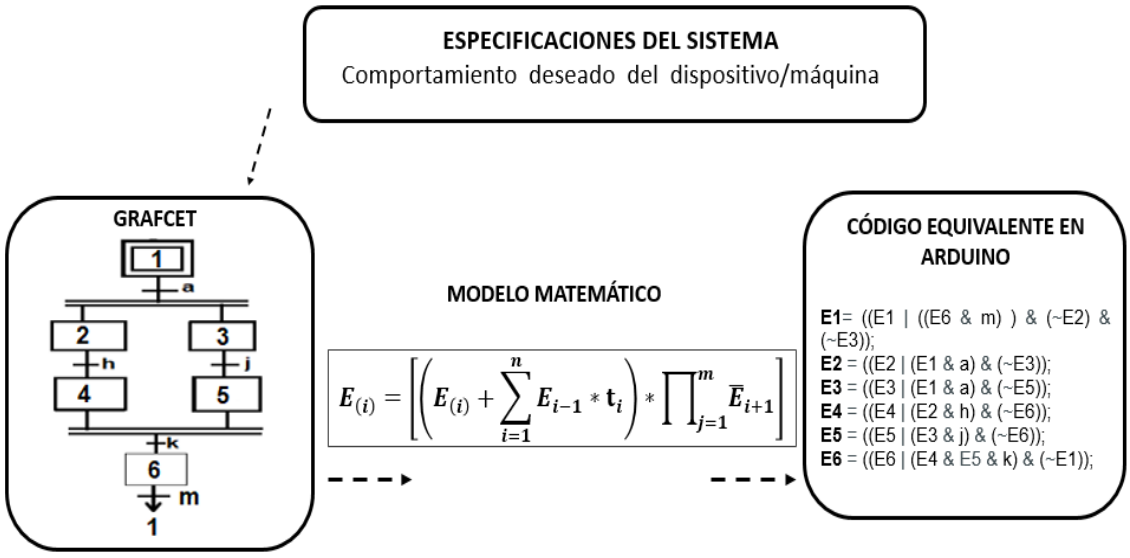

Figura 3. Modelo propuesto

2.4 Ciclo de ejecución del programa en Arduino

La estructura del programa a cargar al procesador de Arduino, se acoge a las recomendaciones sugeridas por la norma IEC 60848, la cual propone organizar el código de programación en tres (3) secciones, así:

Una primera sección llamada TRATAMIENTO *PREVIO* que comprende la declaración de variables asociadas a etapas (Ei), entradas (Si), salidas (Yi), memorias (Mi), temporizadores (Ti) y contadores (Ci), y el void *setup* (), donde se declaran los tipos de pines de entrada y salida como (INPUT u OUTPUT), la definición del estado inicial de las etapas en el primer ciclo del programa y el establecimiento de los pines de salida en estado (0).

A continuación el *TRATAMIENTO SECUENCIAL* inicia con la finalización del *void setup()* y empieza con la declaración del *void loop()*, que tienen cargadas las instrucciones que se ejecutan hasta completar el ciclo, entre ellas la lectura de los puertos de entrada y su asignación a las variables correspondientes, la evaluación del estado de las variables en cada etapa y la ejecución de las acciones en la sección *TRATAMIENTO POSTERIOR*, este esquema del ciclo de ejecución del programa en Arduino se aprecia en la Figura 4.

El asunto de las temporizaciones se soluciona con el uso de dos (2) subrutinas en la que se usa el parámetro *"millis"* del procesador para hacer una comparación de variables, que dan como resultado la activación o desactivación de la variable "Ti".

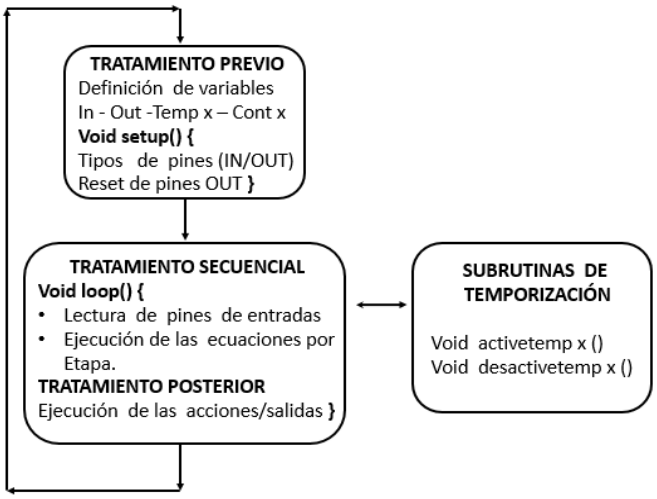

Figura 4. Ciclo de ejecución del programa en Arduino

2.5 Ejemplo de aplicación

Este caso escogido para la implementación de la metodología describe la secuencia de un circuito electroneumático programado para realizar 3 ciclos de salida y entrada del cilindro con solo accionar el pulsador CX3 (Figura 5).

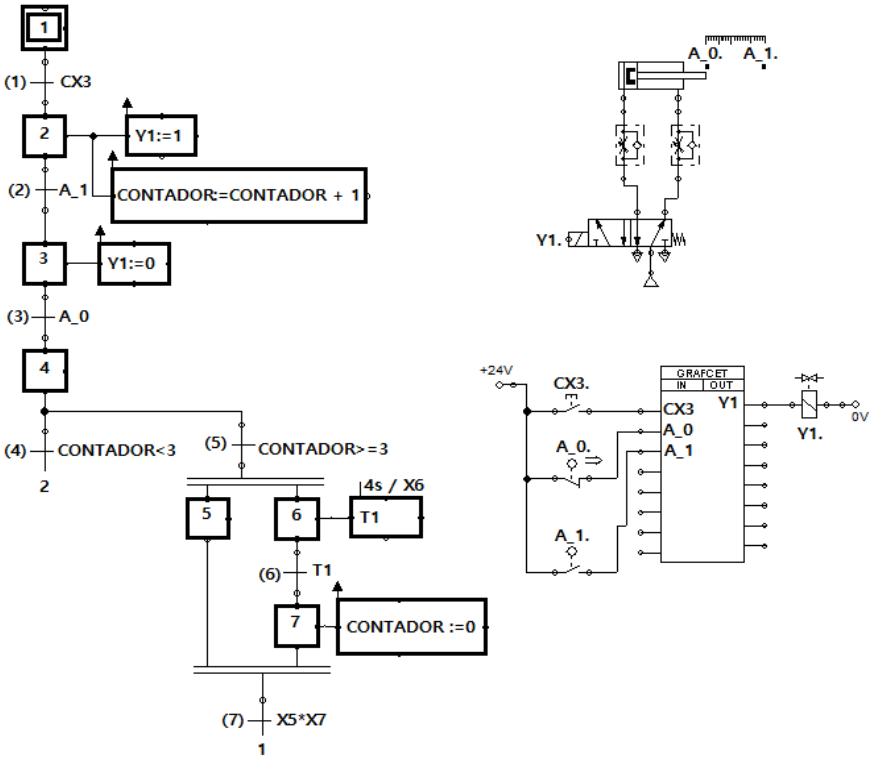

Figura 5. GRAFCET de estudio

Aunque sencillo, este GRAFCET incorpora los elementos fundamentales usados en los proyectos de automatización, como entradas y salidas digitales, temporización, conteo de eventos, y las tres estructuras básicas de los sistemas secuenciales.

2.6 Código equivalente de Arduino

```
// Tratamiento previo
//Declaración de variables utilizadas en el GRAFCET
int E1; int E2; int E3; int E4; int E5; int E6; int E7;// Declaración de las variables asociadas a las etapas
int CX3 = 8;int A_0 = 10; int A_1 = 11;// Declaración de las variables/pines de entrada
int Y1 = 2; // Declaración de las variables/pines de salida

// Variables asociadas al "TEMPORIZADOR 1".
int T1 = 0; // Bit asociado al temporizador 1
int tiempo1 = 4000; // Temporización de 4 segundos
int activado1 = 0; long inicio1, final1, actual1; // Variables usadas en la subrutina de temporización

// Variables asociadas al Contador
int CONTADOR = 0; // Variable que acumula el número de eventos/ciclos
const int PSCONTADOR = 3; // Preselect del contador o # de ciclos deseados.
int ESTADOPREVIO_E2 = 0; // Estado previo de la etapa que incrementa el contador

void setup () {
//Declaración del tipo de puerto digital
pinMode (8, INPUT); pinMode(2, OUTPUT);

//Resetear los puertos/pines de salida en 0v.
digitalWrite(Y1,0);

//Declaración del estado inicial de las etapas
E1 = 1; //La Etapa 1 es una Etapa inicial,
E2 = 0; E3 = 0; E4 = 0; E5 = 0; E6 = 0; E7 = 0; // las etapas comunes inicialmente en 0} // Fin del void setup

void loop () {
// Tratamiento secuencial
//Lectura de valores de puertos digitales de entrada
CX3 = digitalRead (8); A_0 = digitalRead (10); A_1 = digitalRead (11);

// Ecuaciones correspondientes a cada etapa
E1 = ((E1 | ((E5 & E7) & (E5 & E7))) & (~E2));
E2 = ((E2 | (E1 & CX3) | (E4 & (CONTADOR < PSCONTADOR))) & (~E3));
E3 = ((E3 | (E2 & A_1)) & (~E4));
E4 = ((E4 | (E3 & A_0)) & (~E2) & (~E5) & (~E6));
```


```
E5 = ((E5 | (E4 & (CONTADOR >= PSCONTADOR))) & (~E1));
E6 = ((E6 | (E4 & (CONTADOR >= PSCONTADOR))) & (~E7));
E7 = ((E7 | (E6 & T1)) & (~E1));
// Tratamiento posterior Ejecución de las acciones en las etapas
if (E2 == 1) {digitalWrite (Y1, 1);}
if (E2! = ESTADOPREVIO_E2) {if (E2 == 1) {CONTADOR++;} ESTADOPREVIO_E2 = E2;}
if (E3 == 1) {digitalWrite (Y1, 0);}
if (E6 == 1) {activetemp1();} else {desactivetemp1();}
if (E7 == 1) {CONTADOR = 0;}
} // Fin del void loop

// Subrutina temporizador 1
void activetemp1() {
if (activado1 == 0) { activado1 = 1; inicio1 = millis(); final1 = inicio1 + tiempo1;}
actual1 = millis();
if (activado1 == 1 && (actual1 > final1)) {T1 = 1;} else {T1 = 0;} }
//.....
void desactivetemp1() {T1 = 0; activado1 = 0; inicio1 = 0; final1 = 0; actual1 = 0;}
```

3. Resultados

El método mostró su efectividad al ser aplicado al caso de estudio y a múltiples sistemas electromecánicos en entorno de laboratorio, que fueron modelados en GRAFCET haciendo uso de estructuras secuenciales lineales, alternativas, simultaneas, entradas y salidas digitales, además de temporizaciones y contadores.

En este modo de programación, no se requirió ninguna librería de Arduino ni tampoco la compra o el uso de software adicionales al del simulador de GRAFCET y el IDE de Arduino. Como se observa en la Figura 4, el ciclo de ejecución del programa en Arduino es similar a otras formas de programación, caracterizadas por la definición y configuración de las variables, la ejecución del programa principal y la ejecución de las acciones, con el uso complementario de funciones para la temporización.

En esta propuesta, la formulación de las ecuaciones características de cada etapa fue elaborada por el programador sobre la base de la ecuación del binodo mostrada en la sección 2.2.

El código usado para la programación del contador de eventos, resultó ser confiable, al lograr que el número de ciclos incrementara una sola vez por cada disparo de la señal de conteo ascendente, y debido a una división de las funciones incremento, comparación y reseteo del número de ciclos. La programación de la temporización, fue óptima al usar el comando “millis” dentro de una subrutina de temporización y renunciando al uso del comando “delay”, cuyo uso congela el procesamiento del programa durante su ejecución.

Este método ha sido probado en la enseñanza de la asignatura "Diseño de sistemas mecatrónicos" de IU ITSA, en seis (6) grupos de estudiantes en los últimos 2 años, presentando un alto nivel de asimilación al aplicarlo a la programación de equipos MPS (Modular Producción Sistemas).

4. Discusión y conclusiones

Fue posible llevar a la implementación física en un controlador digital de Arduino, la lógica plasmada en GRAFCET, apoyándose en un modelamiento matemático claro y sin ambigüedades, respondiendo a las necesidades de estudiantes o desarrolladores que requieren de una estructura de programación confiable y sobre un hardware de bajo costo.

Queda probado que cualquier automatismo planteado en GRAFCET con los elementos usados en este trabajo, puede ser traducido a una placa Arduino sin pérdida de información.

El proceso para la obtención del código, resulta transparente al aplicar el modelo matemático que describe el estado de activación o desactivación de cada etapa expresado por la ecuación (2).

Todo el razonamiento y desarrollo de la lógica aplicada a la solución del sistema de automatización debe desarrollarse en la fase previa que culmina con el planteamiento de un GRAFCET que cumpla con las especificaciones de desempeño deseadas, a partir de ese punto solo se requiere seguir de manera procedimental, las indicaciones planteadas en la sección 2 de este trabajo para obtener un programa que pueda ejecutarse en el microcontrolador Arduino. Esto implica que no es necesario tener conocimientos avanzados en programación para lograr soluciones de automatización. El automatizar un proyecto GRAFCET sobre Arduino, hace posible la supervisión del programa durante su ejecución, al mostrar en todo momento el estado de sus etapas tanto activas como inactivas.

Esta propuesta ha descrito una potente combinación de tecnologías suficientemente robustas tanto en hardware como en software para afrontar la automatización de sistemas secuenciales de eventos discretos, al implementar los automatismos elaborados en GRAFCET sobre la económica plataforma Arduino.

Esta forma de programar placas Arduino, potencia la realización de proyectos de desarrollo tecnológico, donde un PLC no sería una opción viable por su elevado costo.
Se pudo verificar que un GRAFCET lineal de dos (2) etapas como el requerido para describir la acción de Stuart / Stop en un motor, no funciona bajo este modelo, debido a la definición matemática de las mismas etapas, en la que una etapa no puede estar al mismo tiempo activa e inactiva, requiriendo en este caso un número mínimo de tres (3) etapas para este caso. Por consiguiente, tampoco es correcto hacer un salto justo hacia una etapa precedente, siendo conveniente agregar una etapa de apoyo entre la etapa actual y la etapa de destino. Este trabajo solo abarca la implementación de una fracción de la norma IEC 60848, delimitado a aplicaciones hechas con elementos descritos en el caso de estudio de la sección 2.5.
8

Aunque el lenguaje "Programming" usado para Arduino es ampliamente conocido por los desarrolladores, aún no es de completa aceptación en las aplicaciones de automatización industrial, en parte porque los PLC`s se han ajustado a la norma internacional IEC 6131-3 que ofrece 5 lenguajes de programación.

El desarrollo de esta metodología por el momento debe ser codificada por un programador siguiendo los pasos mencionados en este trabajo, sin embargo, es el insumo principal para la creación de un software que permita la creación del GRAFCET y su generación automática del código.

A partir de este trabajo se podría avanzar hacia la implementación de proyectos de automatización más complejos que incluyan más estados de la guía GEMMMA, donde la programación parte de un GRAFCET maestro que coordina la ejecución de subprogramas con funciones específicas.

5. Referencias

- Schumacher, F. (2015). Transformation of GRAFCET-Based Control Specifications Into an IEC 61131-3 Implementation. M. Eng. thesis, Hamburg University of Technology (TUHH), Hamburg, Germany.
- Schumacher, F., Schröck, S., & Fay, A. (2013). Transforming Hierarchical Concepts of GRAFCET into a Suitable Petri Net Formalism. MIM.
- Schumacher, F., & Fay, A. (2014). Formal representation of GRAFCET to automatically generate control code. Control Engineering Practice, 33, 84-93.
- Giraldo, J.A., Duque, J., & Forero, J.D. (2019). Development of a Tool for Automatic Generation of GRAFCET Control Code Using Arduino IDE. International review of automatic control, 12, 210.
- Philippot, A., & Tajer, A. (2010). From GRAFCET to Equivalent Graph for synthesis control of discrete events systems. 18th Mediterranean Conference on Control and Automation, MED'10, 683-688.
- Lee, G., Han, Z., & Lee, J.S. (2004). Automatic generation of ladder diagram with control Petri Net. Journal of Intelligent Manufacturing, 15, 245-252.
- Villena, R. (2001). Algoritmos y teoría para el diseño y programación de sistemas de control industrial materializados con PLC's. PhD. Eng. thesis, Universitat Politècnica de València, Valencia ,España.
- Fernández, A.R. (1978). Síntesis y análisis de los sistemas digitales secuenciales mediante la teoría binodal. Revista de informática y automática. vol. 11, N°. 35-36, p 16-25.
- Borges, P., Machado, J., Villani, E., & Campos, J.C. (2010). From SFC Specification to C Programming Language on the Context of Aerospace Systems Control. IFAC Proceedings Volumes, 43, 46-51.
- IEC 60848:2013. International Electrotechnical Commission. GRAFCET specification language for sequential function charts -. Geneva, 2013-02.
- IEC 61131-3:2013. International Electrotechnical Commission. Programmable controllers - Part 3: Programming languages -. Genève, 2013.
- Provost, J., Roussel, J., & Faure, J. (2011). A formal semantics for Grafcet specifications. 2011 IEEE International Conference on Automation Science and Engineering, 488-494.
- Bayó-Puxan, O., Rafecas-Sabaté, J., Gomis-Bellmunt, O., & Bergas-Jané, J. (2008). A GRAFCET-compiler methodology for C-programmed microcontrollers. Assembly Automation, 28, 55-60.

Sobre los autores

- **Jovanny Duque**: Ingeniero mecánico,Especialista en Automatización, Máster en Ingeniería de Procesos de la Universidad del Norte. Profesor titular. jduque@itsa.edu.co
- **Jorge Duarte**: Ingeniero mecánico, Doctor en ingeniería mecánica de la Universidad del Norte. Profesor titular. jorgeduarte@mail.uniatlantico.edu.co
- **Rafael Ramírez**: Ingeniero mecánico, aspirante al Doctorado en ingeniería energética de la Universidad de la Costa. Profesor titular. rafaelramirez@mail.uniatlantico.edu.co

---